UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Titulació:

Grau en enginyeria de Sistemes Audiovisuals

Alumne:

Ariadna Xicota Matencio

Enunciat TFG:

Projecte de disseny i desenvolupament d'una aplicació d'agenda de concerts

Director del TFG:

Pau Fernandez Duran

Convocatòria de lliurament del TFG:

Juny 2019

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Projecte de disseny i desenvolupament d'una aplicació d'agenda de concerts

Escola Superior d'Enginyeries Industrial,
Aerospacial i Audiovisual de Terrassa

Grau en enginyeria de Sistemes Audiovisuals

Autor: **Ariadna Xicota Matencio**

Director del TFG: **Pau Fernandez Duran**

10 de Juny de 2019

Índex de continguts

1. Introducció

1.1. Abstract

Resum

Aquest projecte de fi de grau pretén desenvolupar una aplicació Android que permeti a l'usuari accedir, de forma senzilla, a una agenda de concerts recollida de l'entorn web i que permeti fer-ne cerques segons diferents criteris. Per fer-ho possible s'han programat diferents elements: un *scraper*, una base de dades, una API i una aplicació. D'aquesta manera s'ha realitzat una aplicació *fullstack* posant en joc molts dels aprenentatges de programació adquirits durant el grau.

Paraules clau: android, concerts, aplicació mòbil, java, php.

Abstract

This end of degree project develops an Android Application. This Application allows the user to easily access to a concerts' agenda, from the internet, and to search through it using different criteria. In this process, different elements have been programmed: a scraper, a database, an API and the application itself. In conclusion ,a lot of the learnings from the degree have been applied to develop a fullstack application.

Key words: android, concerts, mobile application, java, php.

1.2. Declaració d'honor

I declare that, the work in this Degree Thesis is completely my own work, no part of this Degree Thesis is taken from other people's work without giving them credit, all references have been clearly cited,

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary actions by The Universitat Politècnica de Catalunya – BarcelonaTECH.

Ariadna Xicota Matencio  10/06/2019

Student Name Signature Date

Title of the Thesis : Projecte de disseny i desenvolupament d'una aplicació d'agenda de concerts

1.3. Objecte del treball

L'objectiu principal del treball és desenvolupar una aplicació que permeti a l'usuari accedir de forma senzilla a una agenda de concerts recollida de l'entorn web i fer-ne cerques segons diferents criteris.

1.4. Abast del treball

- Programació d'un s*craper* per recollir informació de l'entorn web.
- Creació d'una base de dades on guardar la informació recollida.
- Programació d'una API web encarregada d'extreure resultats de la informació recollida.
- Configuració d'un servidor web on emmagatzemar i executar el programari desenvolupat.
- Creació d'una aplicació per a dispositius Android amb les següents funcionalitats:
    - Mostrar una llista de futurs concerts ordenats cronològicament.
    - Mostrar informació completa de cadascun d'aquests concerts.
    - Buscar concerts per artista, població i/o data.
    - Guardar i mostrar concerts preferits de l'usuari.
    - Mostrar la localització d'un concert.

1.5. Requeriments del treball

Per desenvolupar aquest projecte s'han marcat uns requeriments als que ha de donar resposta l'aplicació que s'elabora com a producte final, són aquests:

1. Disposar d'una llista de concerts al mòbil.
2. Filtrar concerts seguint diferents criteris triats per l'usuari.

3. Guardar una llista de concerts favorits.
4. Localitzar la situació de cada concert en un mapa.

Per desenvolupar cadascun dels requeriments anteriorment comentats, s'han desenvolupat els següents sistemes, amb diverses funcions:

- *Scraper*:
    - Recollir amb l'*scraper* la informació d'una pàgina de tercers amb la llista de concerts.
    - Connectar l'*scraper* amb la base de dades i omplir-la amb la informació.
    - Llegir amb l'*scraper* la informació de cada pàgina de concerts i guardar-la a la base de dades.
- Base de dades:
    - Generar un base de dades on emmagatzemar tota la informació recollida de l'entorn web.
- API web:
    - Programar l'API per que ens retorni la llista amb els concerts.
    - Programar l'API per que retorni la informació d'un concert.
    - Programar l'API per que retorni una llista de concerts filtrada.
- Servidor:
    - Contractar un servidor a un proveïdor *cloud* (5$/mes).
    - Pujar l'API, la base de dades i l'*scraper* al servidor i connectar-hi l'aplicació.
- Aplicació:
    - Crear una aplicació Android.
    - Mostrar a l'aplicació el llistat de concerts.
    - Programar un filtre per a l'aplicació.
    - Programar l'emmagatzematge de concerts a l'aplicació i la mostra d'aquests en una llista.
    - Mostrar la localització de cada concert en un mapa.

Aquesta aplicació no es publicarà a Google Play (pels drets d'autor del contingut) i per la falta de drets sobre el contingut de tercers, però si aquests es tinguessin, seria necessària una llicència de desenvolupador Android (30$).

1.6. Planificació del treball

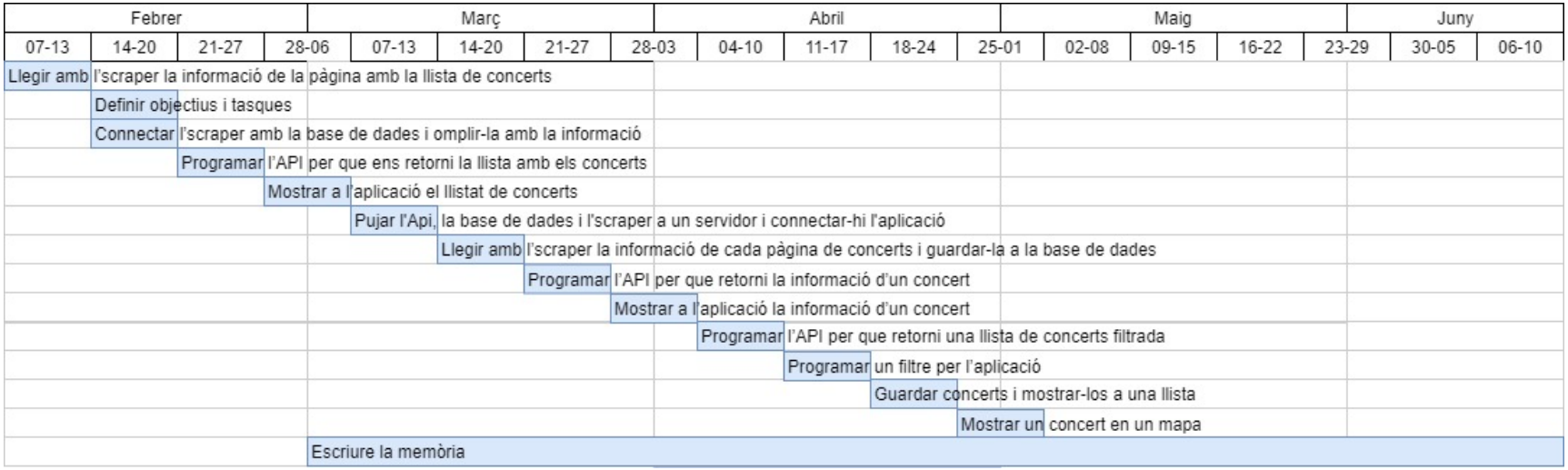


Figura 1. Diagrama de gantt del projecte.

1.7. Utilitat del treball

A Catalunya hi ha una escena musical molt variada, on artistes autòctons i estrangers es reparteixen per actuar per tot el territori cada dia. Mantenir-se informat de tots aquests concerts és una tasca difícil perquè s'ha d'estar constantment consultant webs de diversos grups, artistes i sales de concerts. Per això és interessant desenvolupar una aplicació on es pugui consultar, des d'un sol lloc, tots aquests concerts i, a més a més, fàcilment des dels nostres telèfons. És a partir d'aquesta necessitat que surt la idea inicial d'aquest projecte.

Però aquest estudi busca anar més enllà i em permet aprofundir, com a desenvolupadora novell, en el coneixement del sistema Android i adquirir nous coneixements, estratègies i recursos per al futur laboral. Així doncs, és una primera experiència amb un projecte propi i individual, on pretenc aprendre del meu sector i conèixer els meus punts forts i aspectes interessants a desenvolupar com a programadora.

2. Desenvolupament

2.1. Antecedents i revisió de l'estat de l'art

Degut a la gran escena musical actual, trobem en aquest context diferents aplicacions mòbils relacionades amb el sector musical i dels concerts. Aquestes, generalment, són destinades a la compra d'entrades, però com a tal mostren certes mancances (per exemple: *Bandsintown Concerts*, *Conciertos y Teatro - Ticketea* i *Ticketmaster España*). Per una banda, aquestes només disposen de la informació dels concerts dels que, l'empresa distribuïdora, en ven les entrades, deixant de banda tots aquells concerts oferts per altres distribuïdores. Per altra banda, al ser aplicacions destinades a la compra d'entrades, no inclouen informació de concerts gratuïts.

Dins d'aquest mateix context, trobem aplicacions amb una oferta d'activitats culturals que inclouen entre elles concerts (per exemple: *Time Out: Discover your city*). Però aquestes es limiten només a grans ciutats i no inclouen aquells concerts realitzats en d'altres àrees del territori.

Per altra banda, amb la mateixa funcionalitat, trobem pàgines web que no disposen d'aplicacions pròpies i que ofereixen informació de diferents concerts dins una àrea territorial (per exemple: *infoconcerts.cat, altaveu.cat* i *agendaconcerts.cat*). Però aquestes no sempre tenen un disseny responsiu, fet que en dificulta la visualització des d'un telèfon mòbil. En d'altres casos, aquestes pàgines web no disposen d'informació agrupada de tota l'oferta pública i obliguen l'usuari a consultar cadascuna de les webs dels diferents artistes o distribuïdores.

2.2. Plantejament i selecció d'alternatives

El desenvolupament del projecte es pot dividir en quatre parts estructurals amb propòsits molt diferenciats.

La primera part pretén recollir la informació referent a diferents concerts, prenent com a font internet. Per tal que aquesta recollida d'informació fos automàtica es va decidir crear un *scraper,* una eina que llegeix el text que conté una pàgina web concreta i permet processar-lo i, si es vol, emmagatzemar-ne la informació que ens interessi.

Per desenvolupar aquest programa es decideix utilitzar el llenguatge Python, perquè és un llenguatge que ja s'havia utilitzat anteriorment durant el grau i del que se'n coneixia el funcionament. Alhora, és un llenguatge molt beneficiós per a la relació amb l'entorn web, ja que al ser un llenguatge molt visual, facilita la modificació en cas d'algun canvi en l'entorn web d'on es recull la informació. Per aquest mateix motiu, es van descartar altres llenguatges alternatius com PHP, C++ o C#, entre d'altres, que no ofereixen aquestes facilitats o no se'n tenia el mateix domini.

Partint del *scraper,* sorgeix una segona part: la base de dades. Degut que el *scraper* necessita un cert temps per recollir la informació i presentar-la a l'usuari, es necessari guardar les dades, evitant així l'espera de l'usuari cada vegada que vol accedir a la informació. Per cobrir aquesta necessitat es tria utilitzar una base de dades MySQL, perquè es coneix de les assignatures realitzades a la universitat. Per altra banda, s'ha triat elaborar una base de dades externa, perquè l'ús de fitxers interns, hauria provocat que l'usuari hagués de descarregar tot el contingut al seu dispositiu i què l'aplicació precisés una actualització cada vegada que s'iniciés. Alhora, l'organització de la informació en una base de dades ofereix les cerques de forma més ràpida i fàcil que en un fitxer.

La tercera part busca la forma de comunicar a l'aplicació la informació emmagatzemada. Per això es desenvolupa una API web que permet consultar la informació recollida i exportar-la en un format accessible per a la futura aplicació. Aquest es decideix programar amb llenguatge PHP, ja que se'n tenia coneixements, és un dels llenguatges més utilitzats en programació web i del que se'n poden trobar gran varietat de llibreries que faciliten la consulta de bases de dades. Aquest ventall de llibreries facilita l'elaboració de l'API web, recorrent al codi obert, i possibilita una tria més específica de la llibreria a utilitzar, que respongui a les necessitats i característiques del projecte.

L'última part pretén fer accessible tota aquesta informació a l'usuari. Per això es concep una nova aplicació Android on es presenta la informació recollida a la base de dades. Aquesta es desenvolupa amb llenguatge Java. Aquest llenguatge és una de les opcions que proporciona Google per desenvolupar aplicacions a la seva plataforma. Entre les opcions de llenguatge que s'oferien des de Google, es va escollir Java perquè és l'únic llenguatge per a programació d'aplicacions mòbils utilitzat en el transcurs del grau.

2.3. Desenvolupament de la o les solucions escollides

2.3.1. Disseny de l'aplicació

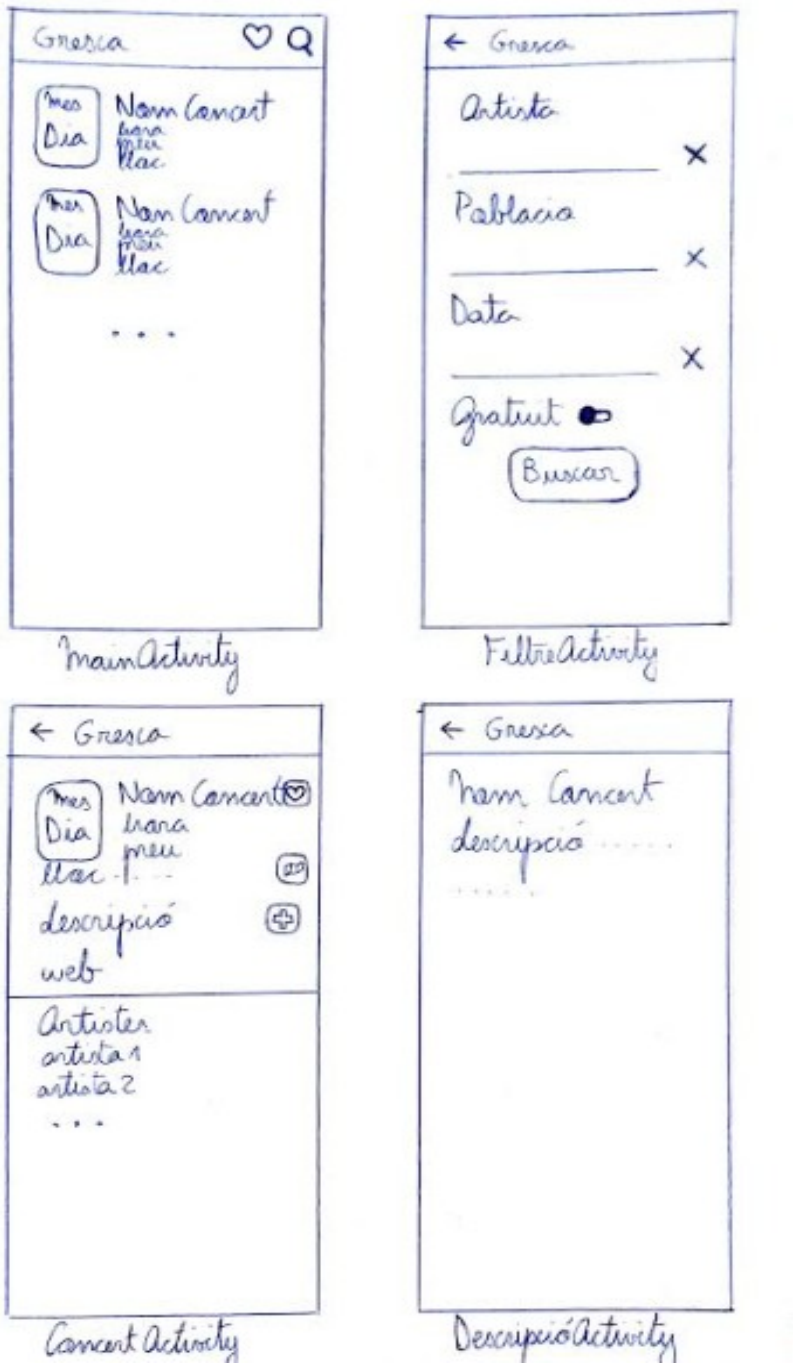


Figura 2. Disseny de les pantalles de l'aplicació.

El disseny de l'aplicació s'ha desenvolupat de manera que fos el més intuïtiu i clar possible. S'han dissenyat quatre pantalles, ja que es van considerar les

mínimes possibles per donar resposta a tots els requeriments. El disseny de cadascuna de les pantalles respon a una distribució ben estructurada, on es diferenciessin clarament les diverses informacions. Per altra banda, s'han escollit l'ús de símbols i/o pictogrames en els botons de cadascuna de les funcions, ja que ofereixen una comprensió ràpida i universal.

Degut que la idea inicial ja incloïa les diferents funcionalitats que s'han desenvolupat al llarg del projecte, no ha calgut fer cap modificació del disseny inicial durant el transcurs del treball. És per aquest motiu que s'ha aconseguit, en l'aplicació final, un disseny molt semblant al proposat, que ofereix a l'usuari una gran facilitat d'ús.

Buscant aquesta claredat i accés intuïtiu, només ha calgut realitzar un canvi en la presentació de la informació de cada concert. Aquesta s'ha organitzat amb un major ús d'icones al proposat inicialment, que ajuden a una lectura i comprensió més clara i ràpida. A nivell estètic, també s'han escollit dos colors molt diferenciats i una tipografia clara, per facilitar-ne la lectura, tenint en compte també situacions de poca il·luminació.

2.3.2. Arquitectura

El sistema està format per 4 parts: una aplicació, una base de dades, un API REST i un *scraper*; dels que es definiran les funcions a continuació.

L'aplicació Android, a la que tindran accés els usuaris i on podran trobar la informació relacionada amb els concerts. En el servidor s'hi troba la base de dades on s'emmagatzema la informació que es mostra a l'aplicació. Perquè aquesta informació sigui accessible, també hi és present una API REST que rebrà peticions HTTP i retornarà la informació necessària en format JSON. La informació mostrada a l'aplicació s'aconsegueix des d'una pàgina web que està situada a internet. Perquè aquesta informació es guardi a la base de dades, es desenvolupa un *scraper*, que serà l'encarregat d'analitzar, estructurar i guardar la informació a la base de dades.

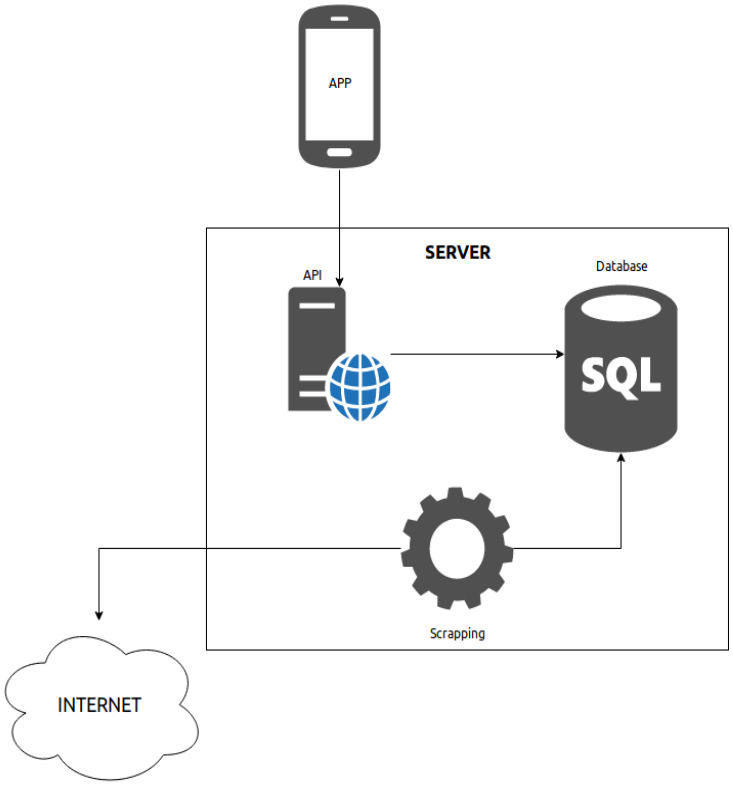


Figura 3. Esquema dels elements del projecte.

Per tal d'organitzar el codi de totes aquestes parts, tenir un control de versions i poder-ho compartir amb el tutor, s'ha utilitzat git i s'ha penjat tot el projecte a GitHub. Git és un software de control de versions que permet crear un projecte i anar guardant diverses revisions dels fitxers. Així, en el moment que s'han acabat uns canvis, es poden registrar les modificacions fetes a cadascun dels fitxers, a la taula d'índexs de git. Una de les altres funcionalitats que permet git és pujar tots els fitxers i la taula d'índexs a un repositori remot, fent possible que qualsevol usuari, amb accés al repositori, pugui veure els fitxers i tots els canvis realitzats des del començament del projecte.

En el cas d'aquest projecte, s'han pujat tots els fitxers a un repositori de GitHub. GitHub és una plataforma propietat de Microsoft que permet tenir repositoris de git públics o privats i veure i descarregar els fitxers i els seus canvis des de la seva pàgina web.

Es pot trobar el codi complet de totes les parts de l'arquitectura a GitHub: https://github.com/ariadnana/projecte

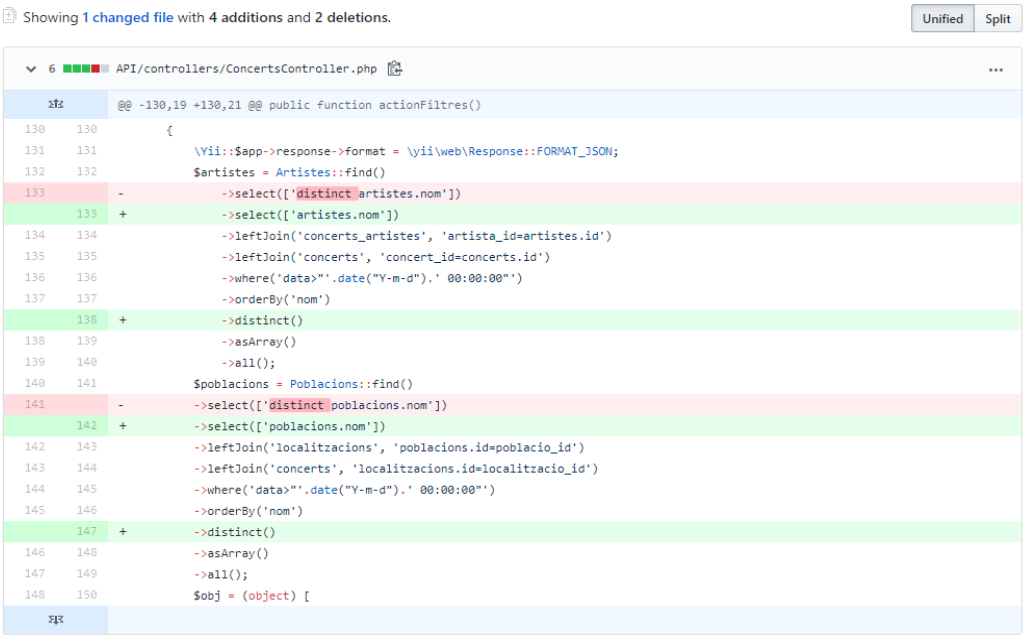


Figura 4. Control de canvis d'un fitxer a GitHub.

2.3.2.1. Scraper

Un *scraper* és una eina desenvolupada per extreure, automàticament, informació disponible a fitxers o pàgines web de la xarxa. L'objectiu d'un *scraper* es basa en extreure d'aquests fitxers continguts de valor per utilitzar-los en d'altres eines.

Així, per exemple, tenim el cas de Google que va desenvolupar eines de *scraping* per llegir pàgines web i les etiquetes de metadades d'aquestes per tal de classificar-les i que els usuaris d'internet tinguin accés a elles. Els *scrapers* de Google comencen mirant una pàgina web qualsevol, en recullen les metadades per classificar aquesta primera pàgina i afegeixen, en una cua, tots els enllaços trobats en el codi HTML, per anar trobant noves pàgines a analitzar. A continuació, segueixen amb les següents pàgines de la cua i així infinitament. Cada cert temps es torna a les pàgines analitzades anteriorment per actualitzar la informació i trobar nous enllaços.

En el cas d'aquest projecte, s'ha desenvolupat un *scraper* que comença consultant un llistat de concerts en una web. A partir d'aquí en guarda les dades i els enllaços de la informació específica que consulta sistemàticament per obtenir tota la informació. Aquest *scraper* repeteix el procediment cada nit, per actualitzar les dades dels concerts ja guardats i trobar-ne de nous.

Per tal d'implementar una eina de *scraping,* he utilitzat el llenguatge de programació Python. Aquest és un llenguatge de programació interpretat, orientat a objectes i d'escriptura dinàmica. És un llenguatge que pretén aconseguir una sintaxi llegible, que faciliti l'aprenentatge i el manteniment del codi. Existeixen diverses llibreries programades amb i per a llenguatge Python, amb l'objectiu d'obtenir el contingut de pàgines HTML (*Beginner's Guide to Python*, 2019).

Per tal d'aconseguir el codi HTML d'una pàgina web, és necessari que Python la descarregui. Per realitzar aquesta descarrega s'utilitza la llibreria Request, que envia una petició HTTP al servidor on està la pàgina amb la ruta indicada aconseguint així el codi font HTML. Una vegada aconseguit el codi, es complementa aquesta funció amb la llibreria LXML que carrega el codi HTML i el converteix en un format de dades estructurat anomenat arbre.[1] Aquesta llibreria permet consultar cada element de l'arbre a partir d'expressions de tipus XPath. Una expressió de tipus XPath permet navegar de forma senzilla per unes dades que estan estructurades amb etiquetes (XML).

```
from lxml import html
import requests


page = requests.get('https://altaveu.cat/concerts?page=1')
tree = html.fromstring(page.content)
ruta = '//div[@class="concerts-list__wrap"]/div'
concerts = tree.xpath(ruta)
i = 1
for concert in concerts:
      data = concert.attrib['data-date']
      nom = concert.xpath(ruta+'['+str(i)+']//h3/text()')
      print(data)
      print(nom[0])
      i=i+1
```

Figura 5. Codi *scraper* LXML.

1 Arbre: Estructura de dades ordenada jeràrquicament que conté tots els elements d'una pàgina web enllaçats entre ells. Així, un element tindrà, com a arrel, l'element que el conté i, com a branques, tots els elements continguts.

El codi de la figura 5 és el primer *scraper* que es va desenvolupar utilitzant la llibreria LXML. El problema, amb l'ús aquesta llibreria, és que tot i haver escollit un element de l'arbre, era necessari escriure repetidament l'XPath des de l'arrel per arribar als elements continguts. Per aquest motiu es busca una altra llibreria que facilités la consulta.

Cercant una nova llibreria amb una sintaxi més adient, es localitza la llibreria Beautiful Soup, que utilitza la mateixa llibreria LXML i estructura d'arbre, però permet trobar elements dins d'un element pare sense haver de reescriure l'XPath des de l'arrel. Amb Beautiful Soup, i un cop has escollit un element pare, pots buscar els diversos elements fills només amb el nom de l'etiqueta d'aquest (*Beautiful Soup Documentation,* s.d.*)*.

```
from bs4 import BeautifulSoup
import requests


page = requests.get('https://altaveu.cat/concerts?page=1')
soup= BeautifulSoup(page.content, 'lxml')
concerts = soup.findAll("div", {"class": "concert-card"})
i = 1
for sibling in concerts:
      data = sibling['data-date']
      nom = sibling.find("h3").text
      print(data)
      print(nom)
      i=i+1
```

Figura 6. Codi *scraper* Beautiful Soup.

El codi de la figura 6 és el primer *scraper* elaborat amb la llibreria Beautiful Soup. Extreu la mateixa informació que el codi de la figura 5, però amb un codi molt més net i fàcil d'actualitzar, si fos necessari. Per això es va decidir utilitzar aquesta llibreria.

Una vegada estructurada tota la informació recollida per l'*scraper*, aquesta s'escriu a la base de dades. Per fer-ho, s'utilitza el *driver* de connexió Pymysql.

Aquest *driver* és l'encarregat de realitzar la connexió entre el programa i la base de dades i executar les comandes SQL per guardar la informació (*Welcome to PyMySQL's documentation!*, s.d.).

Per tal de tenir actualitzada la informació a la base de dades i conèixer els nous concerts, s'ha d'executar l'*scraper* periòdicament. Perquè aquest s'executi automàticament, s'ha programat una tasca de cron que executi l'*scraper* cada dia, a la matinada. Cron es un servei que es troba disponible als entorns de servidors Unix (Linux/BSD i Mac). Aquest servei llegeix un fitxer de configuració i executa les tasques amb la planificació especificada en aquest.

Per iniciar cron, des de la terminal del servidor Linux, situat en qualsevol directori, s'executa la comanda crontab -e. Aquesta comanda obre un fitxer on es pot programar l'execució de qualsevol comanda de terminal, en el moment i/o periodicitat desitjada. Per programar cadascuna d'aquestes tasques, cal escriure la comanda que ha de realitzar amb cinc valors numèrics al davant, que determinen el moment concret que s'ha d'executar. Qualsevol d'aquests valors numèrics pot ser substituït per '*' perquè s'executi en tots els valors possibles dins d'aquell camp. Els diferents valors fan referència a minut, hora, dia del mes, mes i dia de la setmana, en aquests mateix ordre. En aquest cas concret s'han establerts els valors `0 5 * * *', perquè s'executi cada dia a les cinc de la matinada.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 5 * * * python3 /var/www/tfg.xicota.cat/scraper/scraperbs.py >/dev/null 2>&1
```

Figura 7. Codi cron.

2.3.2.2. Estructura de dades

El gestor de bases de dades utilitzat és Mysql, un sistema relacional, multifil i multiusuari que utilitza llenguatge SQL (*MySQL 8.0 Reference Manual. What is MySQL*., s.d.).

L'eix central de l'aplicació és l'entitat 'concerts'. Aquesta entitat conté tota la informació que l'usuari veurà a l'aplicació. A dins del model relacional hi trobem altres entitats relacionades amb 'concerts' que ajudaran que l'usuari realitzi cerques seguint diferents criteris.

L'entitat 'artistes', que conté el nom de l'artista, està relacionada amb 'concerts'. Com que un artista pot participar en diversos concerts i els concerts tenen un o diversos artistes, aquesta relació correspon a una relació N-N. Aquest tipus de relacions a les bases de dades es solucionen amb una taula pivot que conté l'identificador de les taules relacionades.

L'entitat 'localitzacions' té el nom de la sala/espai, una referència a la població on aquesta es troba i una relació amb l'entitat principal 'concerts'. Com que un concert només es produeix en una sala, aquesta relació es una 1-N i es soluciona posant l'identificador de la localització a la taula 'concerts'.

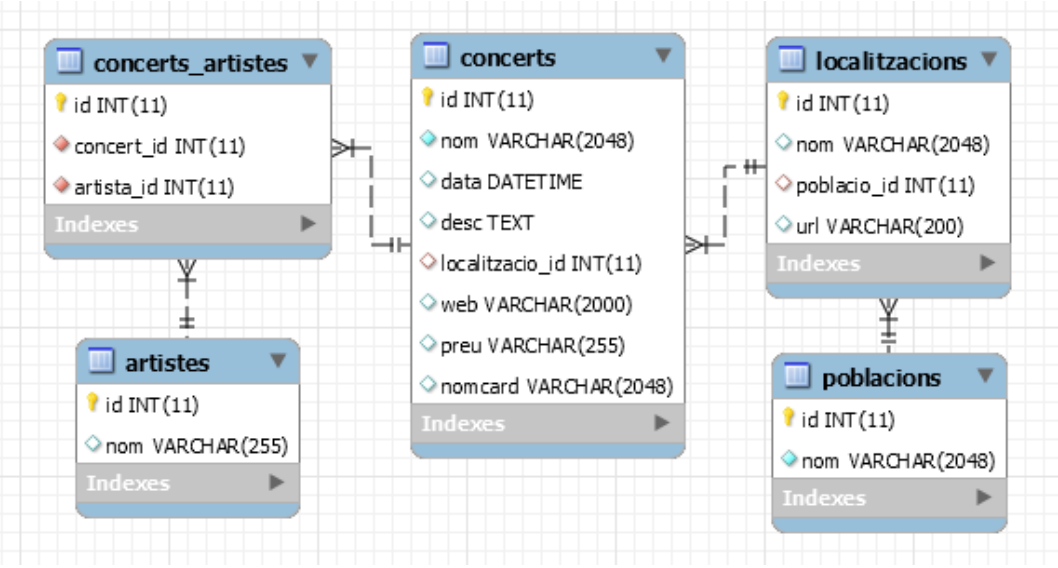


Figura 8. Diagrama entitat-relació de la base de dades.

2.3.2.3. API

L'API és un conjunt de funcions que serveix com a capa de comunicació perquè les interaccions que es realitzen a l'aplicació rebin resposta des del servidor,

tenint en compte les dades emmagatzemades a la base de dades. La modalitat d'API que s'utilitza en aquest projecte és de tipologia *Restful,* que el servidor no s'ocupa de l'estat del client, sinó només de les peticions directes que aquest fa.

Per realitzar la programació d'aquesta API REST s'utilitza el llenguatge de programació PHP i el *framework* Yii2. PHP és un llenguatge de programació interpretat orientat a objectes, molt útil per al desenvolupament web. Aquest llenguatge de programació normalment s'utilitza per proveir contingut dinàmic amb l'ajuda d'un servidor Apache.[2] El codi PHP s'executa al servidor i, amb aquest, es realitzen connexions a bases de dades i a d'altres serveis, per tal de generar codi HTML dinàmic (*¿Qué es PHP?*, s.d.). En el nostre cas utilitzem aquest llenguatge per realitzar connexions a MySQL i generar JSON dinàmics amb el contingut actual emmagatzemat.

Dins del llenguatge PHP trobem moltes llibreries utilitzables i aquestes són de fàcil accés amb el seu gestor de paquets *Composer,* que facilita la baixada i la gestió de les llibreries disponibles. Habitualment trobem aquestes llibreries agrupades en un projecte que les combina, és a dir, un *framework*.

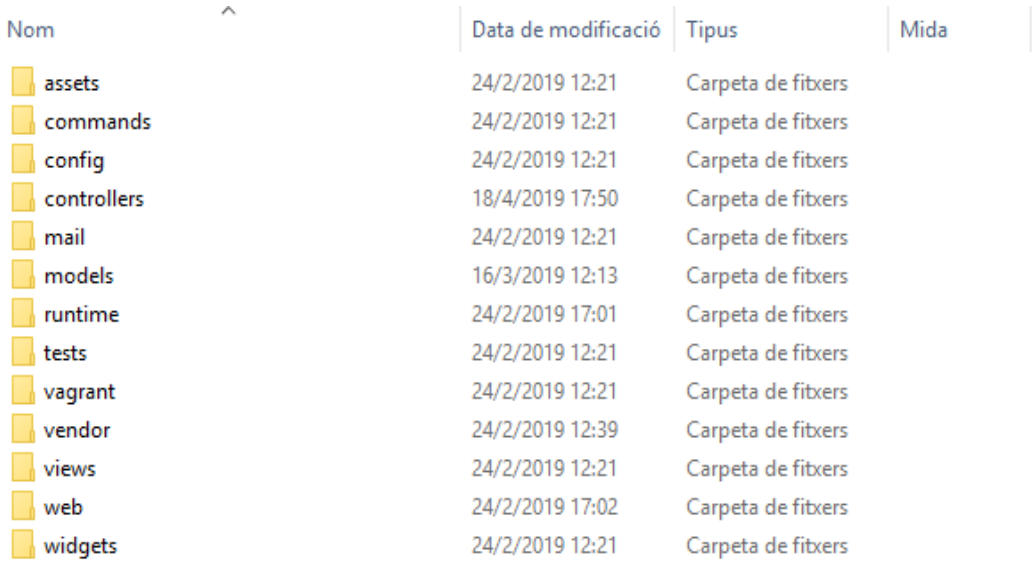
Per realitzar aquesta API es decideix utilitzar el *framework* Yii2, que ja es coneixia prèviament, de la meva feina. Aquest framework destaca per la seva gestió de les bases de dades, amb el patró d'arquitectura del software *ActiveRecord*. Aquest patró indica al codi l'existència dels objectes que estan disponibles a la base de dades. Aquests objectes emmascaren les operacions de cerca, inserció, actualització i eliminació fent que el programador no hagi d'escriure les comandes SQL, i només calgui generar la representació de les entitats al codi (*Introduction: About Yii. The definitive Guide to Yii 2.0. Yii PHP framework.*, s.d.).

Per inicialitzar un projecte amb el framework Yii2 s'ha d'utilitzar una comanda del composer. A l'executar aquesta comanda per terminal es crea, dins la carpeta public, l'estructura bàsica de Yii2.

```
composer create-project --prefer-dist --
stability=dev yiisoft/yii2-app-basic public
```

Figura 9. Comanda per iniciar el projecte de Yii2.

2 Apache és un projecte de codi obert que llança un servei que es comunica amb els clients mitjançant el protocol HTTP.



Nom	Data de modificació	Tipus	Mida
assets	24/2/2019 12:21	Carpeta de fitxers	
commands	24/2/2019 12:21	Carpeta de fitxers	
config	24/2/2019 12:21	Carpeta de fitxers	
controllers	18/4/2019 17:50	Carpeta de fitxers	
mail	24/2/2019 12:21	Carpeta de fitxers	
models	16/3/2019 12:13	Carpeta de fitxers	
runtime	24/2/2019 17:01	Carpeta de fitxers	
tests	24/2/2019 12:21	Carpeta de fitxers	
vagrant	24/2/2019 12:21	Carpeta de fitxers	
vendor	24/2/2019 12:39	Carpeta de fitxers	
views	24/2/2019 12:21	Carpeta de fitxers	
web	24/2/2019 17:02	Carpeta de fitxers	
widgets	24/2/2019 12:21	Carpeta de fitxers	

Figura 10. Estructura bàsica de Yii2.

L'estructura bàsica de Yii2 està composta per:

- Assets: Carpeta on es configuren els fitxers estàtics que carregaran la pàgina web. En el cas de l'API, aquests no s'utilitzen.
- Commands: Carpeta on es programen les funcions que es poden executar des del terminal del sistema operatiu. No s'utilitza a l'API.
- Config: Carpeta on es configura la connexió a la base de dades i les rutes de l'API.
- Controllers: Lògica de programació de les pàgines de l'API.
- Mail: Plantilles dels correus que envia el projecte. No s'utilitza a l'API.
- Models: Definició dels objectes de la base de dades.
- Runtime: Carpeta de fitxers temporals que genera Yii2 a l'executar-se.
- Vagrant: Carpeta on es troba la configuració per arrancar una màquina virtual i executar el projecte. No s'utilitza a l'API.
- Vendor: Carpeta on es troben les llibreries utilitzades.
- Web: Carpeta a la que apunta el servidor per iniciar el projecte.
- Widget: Microcomponents per les llistes. No s'utilitza a l'API.

Una vegada aquesta estructura està generada, es realitza la connexió del projecte a la base de dades a la carpeta config.

```
<?php

return [ 'class' => 'yii\db\Connection',
```

```
      'dsn' => 'mysql:host=localhost;dbname=projecte',
      'username' => 'root',
      'password' => '',
      'charset' => 'utf8',
];
```

Figura 11. Connexió de l'API en Yii2 amb la base de dades.

El *framework* Yii2 funciona amb model-vista-controlador (MVC), que és un patró d'arquitectura del *software* que separa les dades de la lògica de programació de l'aplicació. Per fer-ho possible, el *software* es divideix en tres components: el model, la vista i el controlador.

El model és la representació de la informació en forma d'objecte. Per tant, gestiona l'accés a la informació, tant les consultes, com les actualitzacions. El controlador respon als esdeveniments i utilitza els models quan es rep alguna sol·licitud sobre la informació. I la vista dibuixa i representa el model en la forma desitjada. En el cas de l'API REST es configuren com a respostes de tipus JSON.

En el cas de Yii2, aquest disposa d'un generador de codi que prepara els models a partir d'una estructura de dades. Aquest generador de codi (Gii) utilitza una classe del *framework* anomenada ActiveRecord, aquesta classe és una implementació del patró ORM.[3]

Les classes generades per Gii són utilitzades per obtenir informació estructurada de la base de dades en forma d'objecte. Un cop connectada l'API amb la base de dades, executant el generador de codi i especificant el nom d'una de les taules, aquest et genera el codi del model.

3 *Object Relacional Mapping*: patró que simplifica la comunicació amb la base de dades, eliminant la necessitat d'escriure sentències SQL.

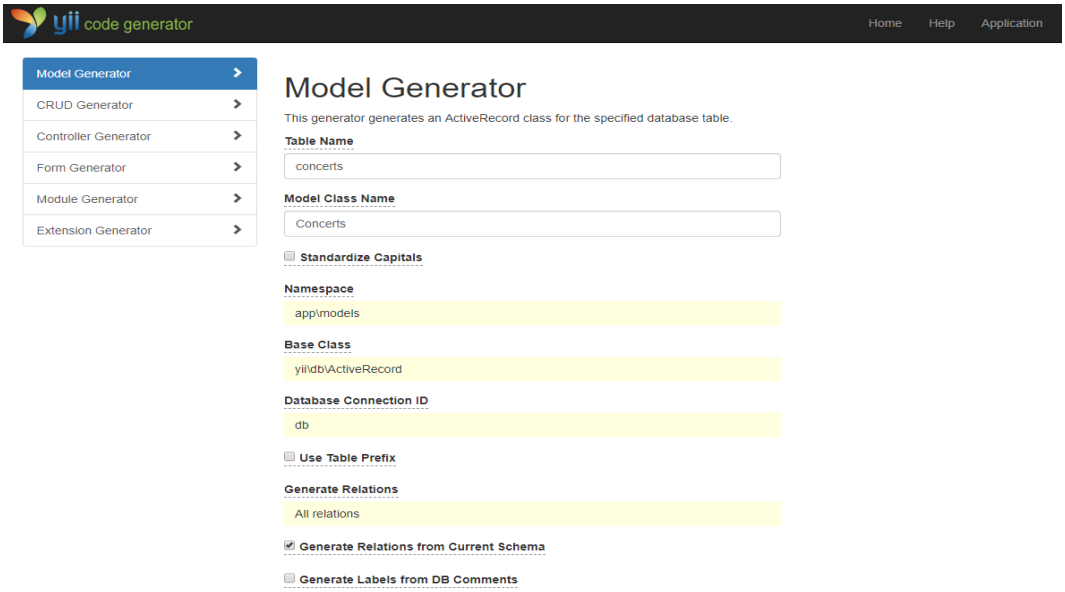


Figura 12. Generador de models de Gii.

El codi generat crea una classe amb el nom de la taula i dins la funció 'tableName' s'especifica l'identificador d'aquesta a la base de dades. Aquest identificador serà, doncs, l'utilitzat en el moment de fer cerques a la base de dades. Aquest codi té la següent forma:

```
class Concerts extends \yii\db\ActiveRecord
{
	public static function tableName()
	{
		return 'concerts';
	}
```

Figura 13. Primera part del codi del model 'concerts'.

La funció 'rules' defineix les regles que han de complir els atributs de l'objecte per poder-lo guardar a la base de dades.

```
public function rules()
{
	return [
		[['nom'], 'required'],
		[['data'], 'safe'],
```

```
            [['desc'], 'string'],
            [['localitzacio_id'], 'integer'],
            [['nom', 'nomcard'], 'string', 'max' => 2048],
            [['web'], 'string', 'max' => 2000],
            [['preu'], 'string', 'max' => 255],
            [
                  ['localitzacio_id'],
                  'exist',
                  'skipOnError' => true,
                  'targetClass' =>
Localitzacions::className(),
                  'targetAttribute' => [
                        'localitzacio_id' => 'id'
                  ]
            ],
      ];
}
```

Figura 14. Segona part del codi del model 'concerts'.

Si la taula està relacionada amb d'altres taules, aquesta genera funcions per trobar l'objecte corresponent d'aquestes altres taules.

```
public function getLocalitzacio()
{
      return $this->hasOne(
            Localitzacions::className(),
            ['id' => 'localitzacio_id']
      );
}
```

Figura 15. Tercera part del codi del model 'concerts'.

Si existeix alguna relació N-N només es genera la funció per aconseguir l'objecte de la taula de relació. Però ActiveRecord disposa de funcions que en faciliten les creacions.

```
public function getPoblacio()
{
      return $this→hasOne(
            Poblacions::className(),
            ['id' => 'poblacio_id']
      )->viaTable(
            'localitzacions',
            ['id' => 'localitzacio_id']
      );
}


public function getArtistes()
{
      return $this→hasMany(
            Artistes::className(),
            ['id' => 'artista_id']
      )->viaTable('concerts_artistes', ['concert_id' => 'id']);
}
```

Figura 16. Última part del codi del model 'concerts'.

Un cop estan generats tots els models, es poden fer cerques fàcilment a la nostra base de dades a partir del mètode find() de Yii2. Així, per exemple, si es volguessin buscar tots els concerts d'avui, es podria aplicar el següent codi.

```
$concerts = Concerts::find()
      ->where([‘data’ => date(“Y-m-d”)])
      ->all();
```

Figura 17. Exemple cerca a la base de dades amb Yii2.

Aquesta línia de codi ens retornaria un vector d'objectes de tipus 'concerts' amb el resultat de la cerca.

2.3.2.3.1. Endpoints

Els endpoints corresponen a les funcionalitats que l'API exposa, és a dir, són els punts d'entrada que utilitza la nostra aplicació per tal de recollir dades o bé realitzar accions al servidor.

Per donar resposta a les diferents funcionalitats de l'aplicació s'han creat 4 *endpoints*: index, concert, filtres i *favs*.

Mètode: concerts.index

```
Petició HTTP:

  GET http://tfg.xicota.cat/concerts/?paràmetres
```

L'*endpoint* 'index' retorna un conjunt de concerts futurs amb la informació formatada necessària per llistar els elements. La informació que cada element té és: id, nom, dia, mes, hora, preu i localització.

A aquest *endpoint* es poden enviar paràmetres opcionals per tal de realitzar un filtratge dels resultats. Aquests paràmetres, degut que utilitzen una petició GET, s'envien per parelles de nom-valor, just després de "?" i separats per "&". En aquest cas concret, els paràmetres que es poden utilitzar són:

- artista, que correspon al nom d'un artista i es buscarà per coincidència exacta.
- data, que correspon a una data en format YYYY-mm-dd.
- població, que correspon al nom d'una població i es buscarà per coincidència exacta.
- gratuït, que correspon a un booleà com a 0 False o 1 True.

Mètode: concerts.concert

```
Petició HTTP:

  GET http://tfg.xicota.cat/concerts/concert/:id
```

L'*endpoint* 'concert' retorna la informació compresa a l'element 'concert' i tota aquella informació de les altres taules que està relacionada amb aquest element. Aquest *endpoint* s'utilitzarà per obtenir la informació individual d'un concert concret. El paràmetre 'id' situat a la URL és obligatori.

Mètode: concerts.filtres

```
Petició HTTP:

      GET http://tfg.xicota/concerts/filtres
```

L'*endpoint* 'filtres' retorna dos llistats: artistes i poblacions. Aquest *endpoint* s'utilitzarà posteriorment per realitzar una predicció en un camp d'escriptura.

Mètode: concerts.favs

```
Petició HTTP:

   GET http://tfg.xicota.cat/concerts/favs/:favs
```

L'*endpoint* 'favs' retorna un conjunt de concerts futurs amb la informació formatada necessària per llistar els elements i filtrats per uns ids, situats a la URL. El paràmetre 'favs' és un conjunt d'ids de concerts separats per comes.

2.3.2.4. Servidor web

Per tal d'allotjar l'API, la base de dades i l'*scraper,* s'ha trobat la necessitat de contractar un servei de *hosting* per tal de donar total disponibilitat per la xarxa a l'aplicació Android. Aquest servidor es troba allotjat a l'empresa Digital Ocean LLC que es gestiona des de la web DigitalOcean.com.

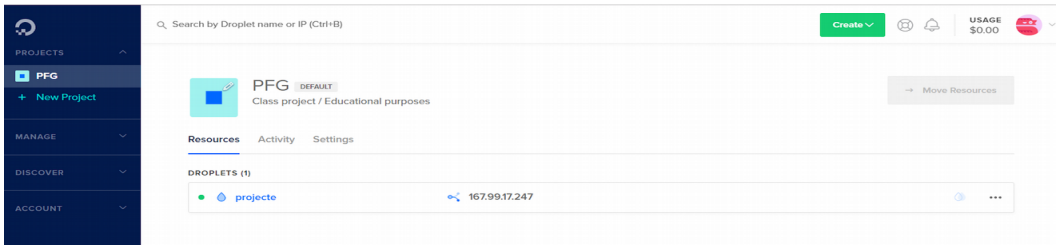


Figura 18. Imatge del panell de DigitalOcean.com.

Digital Ocean LLC és una empresa que es dedica a facilitar servidors *cloud* amb una direcció IP estàtica. A l'hora d'arrencar el servidor ofereixen la possibilitat d'instal·lar un sistema operatiu sense programari o utilitzar una de les imatges preparades amb *software* preinstal·lat.

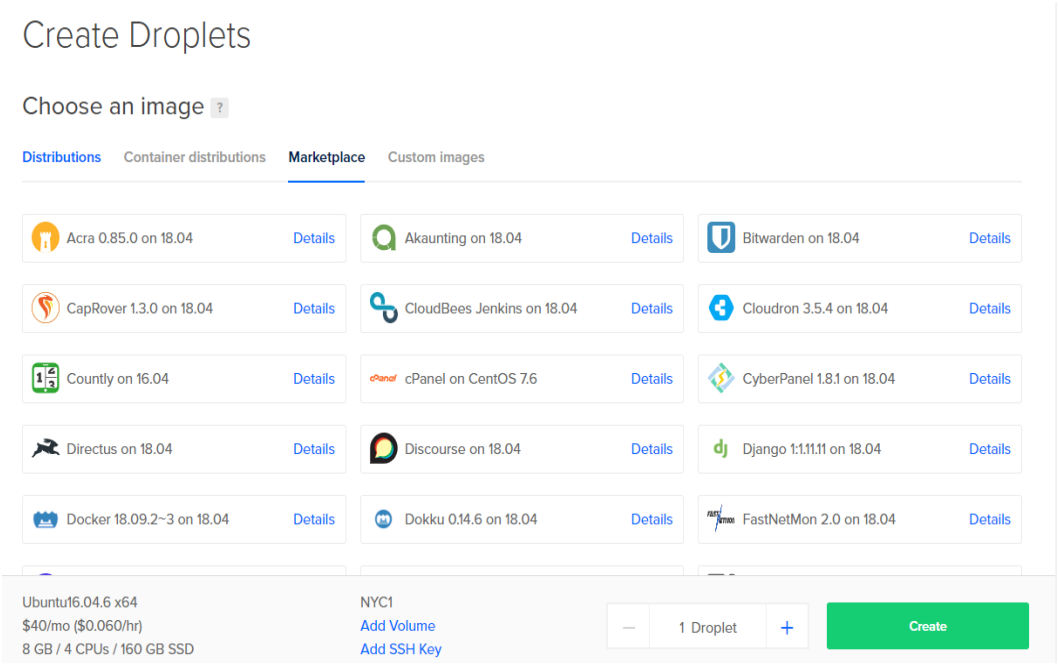


Figura 19. Possibles imatges del servidor.

En el cas d'aquest projecte, s'ha triat utilitzar la imatge LAMP *on 18.04. A*questa imatge correspon a Ubuntu 18.04 amb Apache, MySQL i PHP preconfigurats al servidor. Utilitzant aquesta imatge només és necessari instal·lar Python i així tenir el sistema operatiu a punt per posar en funcionament els components.

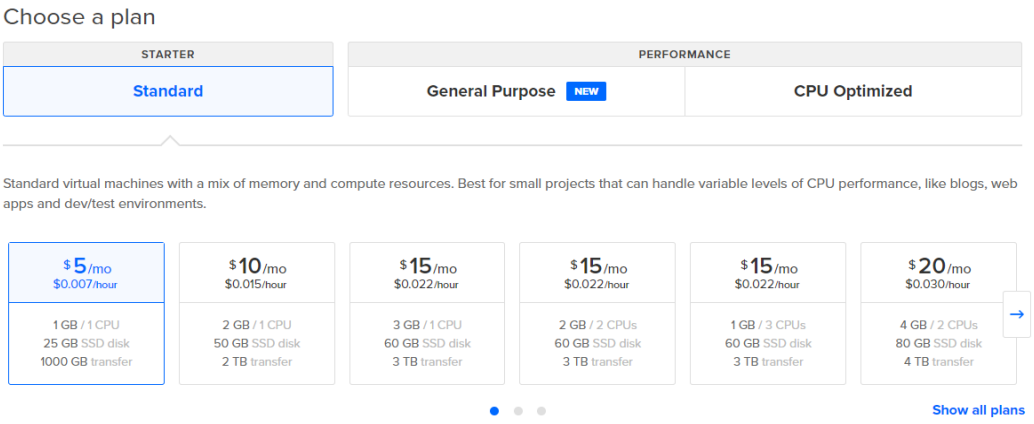


Figura 20. Selecció de capacitat del servidor.

Tot seguit es dóna l'opció de seleccionar la capacitat del servidor. S'escull el servidor més petit ja que, per complir les necessitats del projecte, no en cal un de més gran .

El següent pas necessari és la tria d'on situar físicament el servidor. Normalment, per tal que l'accés dels usuaris sigui el més ràpid possible, s'escull la ubicació més propera al possible públic de l'aplicació. En el cas d'aquest projecte es va decidir situar-lo a Amsterdam.

Finalment, es demana afegir una clau SSH,[4] que s'utilitzarà com a contrassenya per accedir al servidor. Per generar aquesta clau s'executarà la següent comanda al terminal del propi ordinador.

```
ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key
(/home/ariadna/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in
/home/ariadna/.ssh/id_rsa.

Your public key has been saved in
/home/ariadna/.ssh/id_rsa.pub.

The key fingerprint is:

SHA256:Up6KjbnEV4Hgfo75YM393QdQsK3Z0aTNBz0DoirrW+c
ariadna@ordinador

The key's randomart image is:

+---[RSA 2048]----+

|    .      ..oo..|
|   . . .  . .o.X.|
|    . . o.  ..+ B|
|   .   o.o  .+ ..|
|    ..o.S   o..  |
|   o.=. o. . .  .|
|    .oo  E. . .. |

+----[SHA256]-----+
```

Figura 21. Generació de clau SSH.

4 SSH és el nom d'un protocol i programa amb la funció de crear l'accés remot mitjançant un canal segur (encriptat) a un terminal d'un ordinador remot.

Un cop generada la clau SSH, agafem la part pública d'aquesta, que es troba en el fitxer id_rsa.pub, i enganxem el contingut del fitxer en el diàleg de DigitalOcean per crear el servidor.

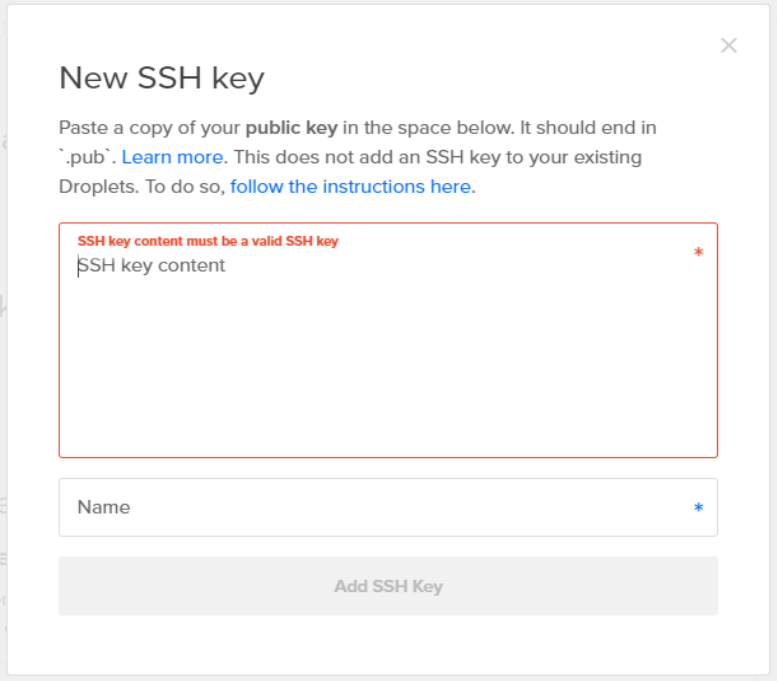


Figura 22. Configuració de la clau SSH al nou servidor.

Tot seguit ja es pot crear el servidor. La plataforma donarà una direcció IP que correspondrà al servidor creat. Amb aquesta direcció IP ja es pot accedir des del terminal de qualsevol l'ordinador que disposi de la clau SSH.

```
C:\>ssh root@167.99.17.247
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Apr 13 15:25:32 UTC 2019

  System load:  0.01              Processes:           102
  Usage of /:   11.9% of 24.06GB  Users logged in:     0
  Memory usage: 45%               IP address for eth0: 104.248.87.85
  Swap usage:   0%

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

71 packages can be updated.
0 updates are security updates.
```

Figura 23. Consola amb accés al servidor.

Es disposava del domini xicota.cat i es va veure convenient assignar el subdomini tfg.xicota.cat a aquest servidor per tal de no utilitzar la IP proporcionada per DigitalOcean LLC. Aquesta configuració es va fer mitjançant el panell de CDmon, empresa amb qui ja s'havia contractat anteriorment el domini xicota.cat.

Figura 24. Panell de control de les DNS de CDmon.

Un cop creat el servidor i configurat el domini per tal de posar en funcionament les diverses parts dels projecte en aquest servidor, es van seguir els següents passos:

- Baixar tot el codi del projecte, clonant el repositori de git.
- Aplicar el codi de creació de la base de dades.
- Canviar el fitxer de configuració de Yii2 perquè l'API accedeixi a la base de dades del servidor en lloc de la base de dades local.
- Crear un fitxer de configuració Apache perquè aquest respongui amb el codi de l'API a les peticions HTTP provinents del domini.

```
<VirtualHost *:80>
        ServerAdmin ariadnaxm@gmail.com
        ServerName tfg.xicota.cat
        ServerAlias www.tfg.xicota.cat
        DocumentRoot /var/www/tfg.xicota.cat/API/web

        <Directory /var/www/tfg.xicota.cat/API/web>
            Options Indexes FollowSymLinks
            AllowOverride All
            Require all granted
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        <IfModule mod_dir.c>
            DirectoryIndex index.php index.pl index.cgi index.html index.xhtml index.htm
        </IfModule>

</VirtualHost>
```

Figura 25. Configuració Apache del servidor.

2.3.2.5. Aplicació Android

L'aplicació mòbil s'ha desenvolupat per dispositius amb sistema operatiu Android.[5] Aquesta aplicació s'ha programat en llenguatge Java, un llenguatge de programació orientat a objectes i compilat per fer-se servir dins la màquina virtual de Java.

Per programar per a Android és necessari instal·lar Android Studio, que té les eines necessàries per compilar i executar el codi en un dispositiu Android.

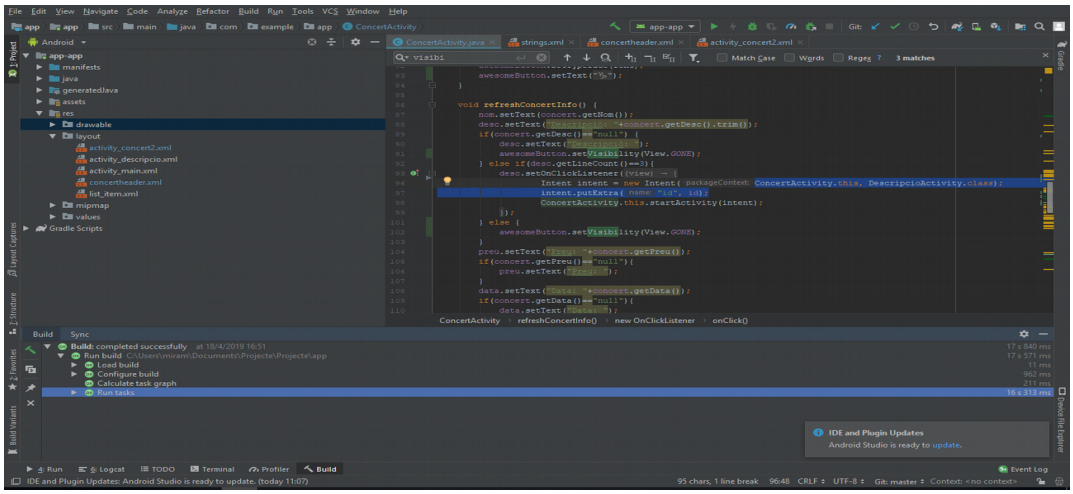


Figura 26. Android Studio.

Per connectar l'aplicació amb l'API s'ha utilitzat Volley, una llibreria programada en Java i per a Android que optimitza les peticions HTTP des de les aplicacions cap a servidors externs. Aquesta llibreria utilitza un sistema de cues que són alimentades per l'aplicació i, un cop resoltes, analitza i prepara el resultat per utilitzar-lo a l'aplicació i, alhora, permet seguir executant codi de l'aplicació sense bloquejar el fil principal, mentre s'espera una resposta del servidor.

```
void fetchConcerts(String url) {
  RequestQueue requestQueue;
  JsonObjectRequest jsArrayRequest;

  requestQueue = Volley.newRequestQueue(this);
  jsArrayRequest = new JsonObjectRequest(
```

5 Android suposava el 88,2 % dels mòbils venuts a Espanya, segons dades de juny de 2018 (*Share of Android operating system sales in Spain January 2016 to June 2018, by month*, s.d.).

```
            Request.Method.GET,
            url ,
            null,
            new Response.Listener<JSONObject>() {
               @Override
               public void onResponse(JSONObject response) {
                    parseJson(response);
                    adapter.notifyDataSetChanged();
                    ProgressBar pgsBar =
(ProgressBar)findViewById(R.id.pBar);
                    pgsBar.setVisibility(View.GONE);
               }
            },
            new Response.ErrorListener() {
               @Override
               public void onErrorResponse(VolleyError error){
                    Log.d(
                        TAG,
                        "Error Respuesta en JSON: "
                        + error.getMessage()
                    );

                    Toast.makeText(
                        MainActivity.this,
                        "Hi ha hagut un error",
                        Toast.LENGTH_LONG
                    ).show();
               }
```

```
        }
  );
  requestQueue.add(jsArrayRequest);
}
```

Figura 27. Codi d'implementació Volley.

Com es pot veure en el codi de la figura 27, la llibreria Volley permet declarar una petició de tipus JsonObjectRequest indicant-li un mètode HTTP, una URL, paràmetres a enviar a la URL i dues funcions a executar si la petició s'executa satisfactòriament o dona un error

Per a cadascuna de les pantalles de l'aplicació s'ha de programar el seu funcionament en dos fitxers. Per una banda, tenim el fitxer on s'escriu el codi Java i, per un altra, el fitxer on s'organitza la vista, estructurat com a XML. Aquests dos fitxers junts formen una *Activity.*[6]

També tenim la representació dels objectes obtinguts des de l'API com a classes independents, per tal de facilitar el tractament d'aquests.

2.3.2.5.1 Estructura de l'aplicació

MainActivity

L'activitat "MainActivity" és la primera que troba l'usuari quan obre l'aplicació. Aquesta activitat és la que utilitza l'aplicació com a llistat, és a dir, altres activitats enviaran paràmetres a aquesta per modificar els concerts mostrats. A l'iniciar-se aquesta activitat es mostra un *spinner,* mentre l'aplicació fa una petició utilitzant Volley a l'*endpoint* 'concerts.index' de l'API. Quan es rep la informació s'amaga el *spinner* i aquesta es mostra en una llista.

6 Una *activity* és un component que conté una pantalla en la qual poden interactuar els usuaris per realitzar una acció. A cada *activity* se li assigna una finestra amb un disseny propi.

Figura 28. MainActivity.

Per mostrar la llista de concerts s'ha utilitzat un RecyclerView, amb una plantilla personalitzada. Un RecyclerView és una versió més flexible d'una llista, que permet personalitzar la plantilla dels ítems propis.

Figura 29. Disseny de la plantilla personalitzada.

L'usuari en aquesta vista pot realitzar les següents accions:

- Si prem la icona de cerca viatjarà a l'activitat "FiltreActivity". Mentre aquesta estigui oberta, l'activitat "MainActivity" es mantindrà a l'espera. Posteriorment rebrà, si s'escau, els filtres seleccionats per l'usuari. Aleshores, consultarà l'*endpoint* 'concerts.index' de l'API enviant aquests filtres com a paràmetres. Un cop rebuda la informació de l'API es mostrarà al llistat i es farà visible un botó que permetrà borrar els filtres i tornar al llistat general.

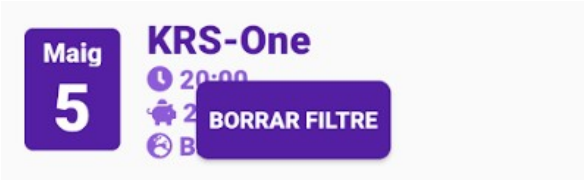


Figura 30. Botó Borrar filtre MainActivity.

- Si prem la icona de favorits, aquesta mateixa activitat consultarà la clau 'favs' a "*SharedPreferences*"[7] que li retornarà el conjunt d'ids dels concerts guardats com a favorits. Amb la cadena rebuda, farà una nova petició a l'API utilitzant l'*endpoint* 'concerts.favs' i llistarà els nous resultats. Quan mostra els concerts favorits, també fa visible un botó per tornar a la visualització anterior.

Figura 31. Botó Mostrar tots MainActivity.

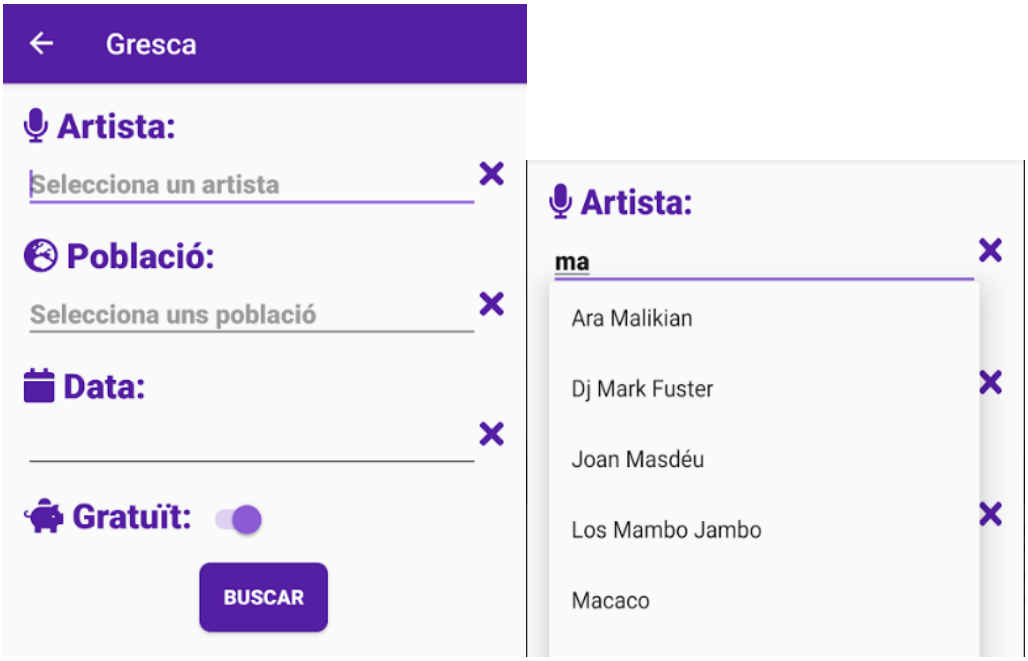
- Si prem en qualsevol dels concerts, va a la 'ConcertActivity'.

FiltreActivity

L'activitat "FiltreActivity" és la que permet a l'usuari realitzar cerques sobre el llistat general de concerts. Quan s'inicia aquesta activitat, es fa una petició a l'*endpoint* 'concerts.filtres' de l'API per omplir les prediccions dels camps d'artista i població. Quan l'usuari comença a escriure en un d'aquests dos camps l'aplicació fa una cerca a les prediccions recollides de l'API i mostra les possibilitats que coincideixen amb el text introduït per l'usuari, perquè les pugui clicar.

Per configurar aquests camps de text amb predicció s'ha utilitzat un AutoCompleteTextView. Per inicialitzar aquest camp se li passa un vector amb valors de tipus text que inclou els possibles resultats. Quan l'usuari comença a escriure, l'aplicació obrirà un menú desplegable amb els valors del vector que puguin coincidir amb el text escrit.

7 *SharedPreferences* és una API que ens proporciona Android per tal de poder guardar dades de forma persistent i així personalitzar l'experiència d'usuari a l'aplicació.



Figures 32 i 33. FiltreActivity.

Quan l'usuari toca al camp de data, l'aplicació invoca el *DatePickerDialog* i aquest mostra un selector de data nadiu d'Android en el que els usuaris d'Android ja estan acostumats a interactuar.

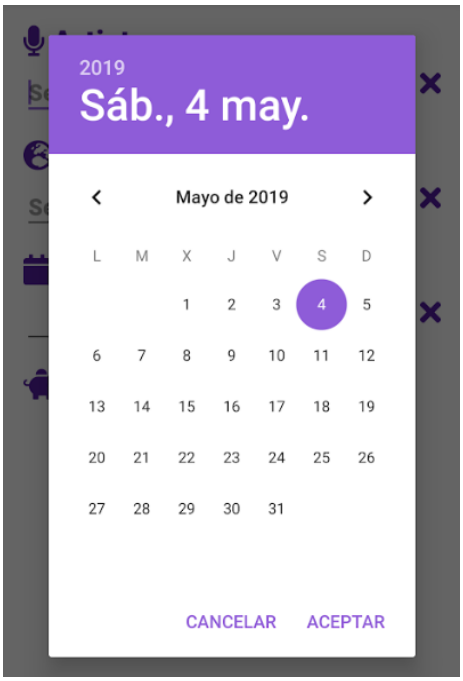


Figura 34. *DatePickerDialog* FiltreActivity.

L'usuari podrà buidar qualsevol dels camps de cerca clicant a la creu situada al costat del camp corresponent.

L'última opció que pot escollir l'usuari és si limitar o no el resultat als concerts gratuïts. Això ho pot seleccionar amb un *widget* anomenat *switch* que té forma d'interruptor.

L'usuari, a part d'introduir totes aquestes dades, pot dur a terme dues accions:

- Clicant a la fletxa de la barra superior, l'aplicació tornarà al llistat anterior.
- Clicant al botó buscar l'aplicació, retornarà els filtres seleccionats per l'usuari a la "MainActivity".

ConcertActivity

L'activitat 'ConcertActivity' mostra tota la informació d'un concert concret. A l'iniciar-se aquesta activitat, es mostra un *spinner,* mentre l'aplicació fa una petició utilitzant Volley a l'*endpoint* 'concerts.concert' de l'API. Quan es rep la informació, s'amaga el *spinner* i es mostra la informació rebuda. Alhora, es consulta la clau 'favs' a *SharedPreferences*. Si aquest concert es troba entre els favorits seleccionats per l'usuari, el botó de favorits es mostra seleccionat (de color blanc). En canvi, si aquest no ha estat seleccionat anteriorment per l'usuari com a favorit, el botó es mostra sense seleccionar (de color lila).

Per poder incorporar la llista d'artistes a aquesta pantalla i que l'usuari es pugui desplaçar per tota la informació, s'ha utilitzat un ListView, on hem afegit com a ítems els artistes del concert. Per mostrar la resta d'informació del concert, s'ha utilitzat una plantilla personalitzada com a capçalera de tota la llista.

Figura 35. ConcertActivity.

L'usuari pot dur a terme diverses accions dins d'aquesta activitat:

- Clicant la fletxa de la barra superior, l'aplicació tornarà al llistat anterior.
- Clicant el botó de favorits, pot introduir-lo o eliminar-lo de la llista de concerts favorits.
- Clicant el botó de localització, s'obre l'aplicació de Google Maps amb la ubicació del concert.
- Si la informació del concert ocupa més de tres línies, es mostrarà un botó lateral d'accés a la informació completa (DescripcioActivity).
- Si en les dades recollides del concert, es disposa d'una URL amb més informació, aquesta apareixerà al final de la pantalla. Clicant en aquest enllaç s'obrirà la pàgina corresponent al navegador seleccionat per defecte.

DescripcioActivity

L'activitat 'DescripcioActivity' mostra tot el text informatiu que s'ha emmagatzemat d'un concert quan aquest ocupa més de l'espai designat per a tal a 'ConcertActivity'. A l'iniciar-se rep el nom i tot el text de la activitat 'ConcertActivity' i la mostra a l'usuari.

Figura 36. DescripcioActivity.

L'usuari pot clicar a la fletxa de la barra superior per tornar a l'activitat 'ConcertActivity' corresponent.

3. Resum de resultats

3.1. Resum del pressupost i estudi de viabilitat econòmica

Com ja s'ha anat explicant durant la introducció i el desenvolupament, la realització d'aquest projecte ha comportat diferents tasques, amb diferents temps de dedicació. Tota aquesta feina comportaria un cost a nivell laboral basant-se en el conveni d'empreses de consultoria. Totes aquestes dades estan detallades a la taula següent.

Activitat	**Duració (hores)**	**Cost/ hora (€/hora)**	**Cost total (€)**
Realització del project charter	5	8,50	42,50
Cerca de possibles servidors i el seu funcionament	10	8,50	85,00
Contractació i configuració del servidor	30	8,50	255,00
Cerca d'alternatives i recursos per desenvolupar l'*scraper*	20	8,50	170,00
Desenvolupament de l'*scraper*	40	8,50	340,00
Disseny de la base de dades	10	8,50	85,00
Desenvolupament de la base de dades	30	8,50	255,00
Cerca d'alternatives i recursos per desenvolupar l'API web	15	8,50	127,50
Desenvolupament de l'API web	80	8,50	680,00
Cerca d'alternatives i recursos per desenvolupar l'aplicació Android	40	8,50	340,00
Disseny de l'aplicació Android	20	8,50	170,00
Desenvolupament de l'aplicació Android	160	8,50	1.360,00
Elaboració dels documents de l'estudi	120	8,50	1.020,00
Total	**580**	**8,50**	**4.930,00**

Taula 1. Cost tècnic de la redacció del projecte.

Tenint en consideració la feina realitzada fins ara i les possibles millores que es podrien realitzar a l'aplicació, es pot valorar necessari un any de treball per al complet desenvolupament del programari. És per aquest motiu que, en el següent pressupost, es planteja una diferència de cost entre el primer any i els següents. Aquesta diferència també ve marcada per aquelles despeses de pagament únic.

Com es pot observar en la taula-resum següent, el pressupost plantejat només inclou despeses, ja que no s'ha programat cap entrada d'ingressos. Això és així degut al seu propòsit acadèmic, que no busca un benefici o intenció mercantil. Aquest fet, produeix un valor negatiu, pel que podem afirmar que aquest producte, en aquestes circumstàncies, no és viable econòmicament.

	Primer any	**A partir del segon any**
Programador	16.531,76 €[8]	8.265,88 €
Servidor	54,00 € (5 $/mes)	54,00 € (5 $/mes)
Llicència Google play	27,00 € (pagament únic)	0,00 €
Ordinador (MacBook 12" 1,2 GHz)	1.500,00 €	0,00 €
Total	**18.112.76 €**	**8.319,88 €**

Taula 2. Resum dels costos del projecte.

3.2. Conclusions

Aquesta aplicació va sorgir de la idea d'unificar la informació de diversos concerts en un sol espai, i fer-la accessible i còmode des dels smartphones. Al finalitzar aquest projecte es pot dir que s'ha aconseguit una aplicació que facilita l'accés a tota aquesta informació. També es pot concloure que, tot i recollir actualment la informació d'una sola font, l'entorn ja és capaç de gestionar informació de múltiples fonts i fàcilment es podria modificar l'*scraper* per possibilitar-ne la recollida.

Quant a la planificació, el desenvolupament ha estat divers. Les primeres fases, a l'utilitzar tecnologies més conegudes i usades, es va reduir el temps necessari per a cada tasca. D'aquesta manera es va agilitzar i es van poder acabar les primeres tasques en menys temps de l'esperat. En iniciar la programació de l'aplicació, en canvi, es van trobar més dificultats. La poca experiència i desconeixement d'algunes de les tecnologies requerides va fer endarrerir les tasques fins a recuperar el calendari planificat inicialment.

En la planificació inicial, es contemplava una evolució paral·lela dels diversos elements de tota l'arquitectura del projecte. Aquesta permetia centrar-se en una funcionalitat cada vegada i veure resultats amb més immediatesa. També va servir per identificar procediments o mètodes a utilitzar en cadascun dels

8 Sou base marcat per conveni (XVII Convenio colectivo estatal de empresas de consultoría y estudios de mercados y de la opinión pública, 2018) per un programador sènior, cap d'operació o programador a internet a jornada completa durant el primer any, per al desenvolupament de l'aplicació, i a mitja jornada a partir del segon any, com a manteniment i revisió del sistema.

elements que facilitaven el funcionament d'altres abans d'haver completat tot un element.

Al decidir la planificació, es va endarrerir una mica l'inici de la memòria per concentrar tot el temps invertit en el desenvolupament de l'aplicació i tot el sistema. Això va facilitar una posterior concentració en la redacció de la memòria un cop acabat tot el desenvolupament tècnic.

Quan es va plantejar un primer disseny, es pretenia mostrar la localització d'un concert en la mateixa aplicació. Per això eren necessàries les coordenades de la localització i inserir el mapa en una de les activitats. El primer obstacle trobat és que l'ús d'aquests procediments a través de l'API de Google Maps té un cost. Un altre inconvenient és que, des del mapa inserit a l'aplicació, no es poden aconseguir indicacions i té menys funcionalitats que l'aplicació de Google. Per aquests motius, finalment es van modificar diversos elements de l'arquitectura per recollir la URL completa del mapa i obrir directament l'aplicació de Google Maps.

En conclusió, malgrat que el resultat final i el desenvolupament del projecte no ha respost de manera idèntica al disseny i planificació inicials; l'aplicació compleix els requeriments previs i els objectius programats.

3.3. Recomanacions de continuació del treball

L'aplicació desenvolupada té un valor acadèmic, i com a primer producte d'elaboració pròpia, mostra una interfície senzilla i primerenca que es podria millorar àmpliament amb més temps i coneixements. En aquest sentit, hi ha tot un seguit d'ampliacions tècniques que es podrien realitzar a l'aplicació per millorar-ne el funcionament i augmentar-ne funcionalitats.

La primera millora previsible, des del punt actual, seria aconseguir més fonts d'informació, augmentant així la quantitat de concerts recollits i diversificant-ne la tipologia. Per aconseguir això caldria desenvolupar variacions de l'*scraper,* per recollír la informació d'altres pàgines web.

Es podria continuar afegint noves funcionalitats a l'aplicació. Entre elles, seria avantatjós la possibilitat de compartir un concert amb amics i/o coneguts a través d'altres aplicacions o a les xarxes socials. Una altra funcionalitat possible seria habilitar la compra d'entrades dins l'aplicació, a través d'un servei de pagament. També seria beneficiós fer un filtratge de concerts per proximitat a l'usuari i mostrar-los en un mapa. Un altre aspecte que es podria millorar, és la creació

d’un registre personal, per tal d’accedir a les dades guardades des de diversos dispositius.

Finalment, per facilitar algunes de les funcionalitats plantejades i arribar a més usuaris, seria una gran millora tècnica desenvolupar l’aplicació per a d’altres sistemes operatius, com iOS.

Per altra banda, cal tenir en compte que, de plantejar-se una evolució futura d’aquesta aplicació, es desenvoluparia més enllà del projecte acadèmic. En aquest sentit, per a una hipotètica continuació del treball, es podria valorar una possible intenció de sortida al mercat. Seguint aquesta idea, caldria buscar la viabilitat econòmica del producte. Així doncs, seria necessari planificar una o diverses fonts d’ingressos per fer viable el desenvolupament i la posada en marxa del producte.

Per fer-ho possible hi ha diverses opcions:

En primer lloc es podria contemplar la possibilitat d’afegir anuncis externs a l’aplicació Android. Per altra banda, es podria establir un cost per l’aplicació que, en concordança amb les descàrregues de l’aplicació, donés resposta a la necessitat econòmica. Altrament, un cop l’aplicació hagués assolit l’èxit desitjat, es podria proposar a diverses plataformes o empreses de mostrar el seus concerts a canvi d’una quota econòmica. Com a darrera opció, i com ja s’ha comentat dins els millores tècniques, es podria vincular la compra d’entrades en la mateixa aplicació, creant un sistema de pagament, del que se’n podria obtenir una comissió proporcional de cada venda.

4. Referències bibliogràfiques

Beginner's Guide to Python. (13 de gener de 2019). Recuperat el 30 de març de 2019 de https://wiki.python.org/moin/BeginnersGuide

Beautiful Soup Documentation. (s.d.). Recuperat el 30 de març de 2019 de https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Welcome to PyMySQL's documentation! (s.d.). Recuperat el 30 de març de 2019 de https://pymysql.readthedocs.io

MySQL 8.0 Reference Manual. What is MySQL. (s.d.). Recuperat el 31 de març de 2019 de https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

¿Qué es PHP? (s.d.). Recuperat el 31 de març de 2019 de https://www.php.net/manual/es/intro-whatis.php

Introduction: About Yii. The definitive Guide to Yii 2.0. Yii PHP framework. (s.d.) Recuperat el 31 de març de 2019 de https://www.yiiframework.com/doc/guide/2.0/en/intro-yii

Share of Android operating system sales in Spain January 2016 to June 2018, by month. (s.d.). Recuperat el 13 d'abril de 2019 de https://www.statista.com/statistics/464221/porcentage-android-smartphone-sales-spain/

XVII Convenio colectivo estatal de empresas de consultoría y estudios de mercados y de la opinión pública, 6 de març de 2018. Recuperat de https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf

5. Índex de figures

6. Índex de taules